Ontario Hydro

RESEARCH DIVISION

91602
new 82-10

ontario hydro
research division

A SIMULATION MODEL FOR RELIABILITY ASSESSMENT OF REPAIRABLE SYSTEMS WITH MAINTENANCE

Report No 86-343-K

D. Bartlett - Student
S.H. Sim - Engineer
Reliability & Statistics Section
Operations Research Department

ABSTRACT

A computer simulation model has been developed for the reliability assessment of repairable multi-component systems under specified operating and preventive maintenance policies. The software package was written in Micro-Soft FORTRAN IV and implemented on an IBM-compatible personal computer. It will be useful for the reliability assessment of repairable systems consisting of N identical subsystems or components. Examples of such systems are boiler-feed pumps of nuclear generating units and coal pulverizers of coal-fired generating units.

job	file	date	report no.
72-605-022	813.08	December 31, 1986	86-343-K

TABLE OF CONTENTS

		Page
1.	INTRODUCTION	1
2.	THE SYSTEM	2
	2.1 System Description	2
	2.2 Assumptions	3
3.	GPMS SOFTWARE PACKAGE	3
4.	SAMPLE APPLICATION	4
5.	FURTHER WORK	6
REFERENCES		7
APPENDIX A - USER'S GUIDE		8
	A1. Introduction	8
	A2. Requirements for Program Execution	8
	A3. Execution Time	8
	A4. Operating the GPMS Program	9
	A5. Sending Output to a Printer Connected to PC	18
APPENDIX B - SUPPLEMENTARY INFORMATION TO USER'S GUIDE		19
	B1. Simulation Run Times and Statistics Collection	19
	B2. Preventive Maintenance Policies	21
	B3. Random Variate Generation	23
	B4. Simulation Output	27
APPENDIX C - PROGRAMMER'S GUIDE		30
	C1. Functional Descriptions	30
	C2. GPMS Program Segments	32
	C3. Description of Routines and Functions	33
	C4. Variables	36
	C5. Program Maintenance Requirements	38
	C6. Miscellaneous Notes	39
	C7. Program Listing	40

ontario hydro
research division

To Dr. D. Mills
Director of Research

A SIMULATION MODEL FOR RELIABILITY ASSESSMENT OF REPAIRABLE SYSTEMS WITH MAINTENANCE

1. INTRODUCTION

Reliability analysis of engineering systems have been made frequently under the assumption that operating times are exponentially distributed. The exponential distribution is favored by reliability analysts because of its tractability. The exponential assumption suggests that preventive maintenance is not required. This may not be valid in practice.

Under the assumption of no preventive maintenance, several probabilistic models have been developed to assess the steady state reliability of a system of identical components [1,2,3,4,6]. A comprehensive literature survey on such systems has been written by Stecke and Aronson [5]. It is interesting to note that Bunday and Scraton [1] have shown that the steady state reliability of a system of independent components with exponential repair times does not depend on the probability distribution of component times to failure, only on their means.

Analytical modeling of multi-component systems with preventive maintenance is very difficult. No significant results have been reported in the literature.

In this report, a computer simulation model has been developed for the reliability assessment of repairable multi-component systems under specified operating and preventive maintenance policies. The software package GPMS (Group Preventive Maintenance Simulation) was written using Micro-Soft FORTRAN IV and implemented on an IBM-compatible personal computer with an 8087 processor. It can generate statistical information from which component availability and repair team utilization can be determined. A transaction log of the system's state throughout the simulation can be produced on request.

2. THE SYSTEM

2.1 System Description

The system consists of N identical components, and it can provide 100% service if at least k ($\leq$ N) components are operating. A total of c repair steams is available. The system is operated under one of the following policies:-

(a) All available components are placed in operation, with each of the components carrying an equal share of the load up to its maximum capacity.

(b) k components are operated to 100% capacity. The remaining operable components are placed on standby; during the standby period, the component cannot fail.

(c) The specified number of components operated is greater than k but less than N, with more than sufficient operating capacity to provide 100% service, and the rest of the components on standby.

One of the following preventive maintenance policies may be applied to the system.

(a) No preventive maintenance.

(b) Group age maintenance policy.

Under this policy, a component may be removed for preventive maintenance after a fixed elapsed time since the completion of the last preventive maintenance event.

(c) Group block maintenance policy.

The starting times of preventive maintenance events are fixed at regular intervals. If any of the maintenance events is delayed due to operating constraints, the starting times of subsequent maintenance events will not be affected.

(d) Individual age maintenance policy.

This refers to a policy of performing preventive maintenance on each component after it has operated for a specified time period.

The following conditions must be satisfied before any preventive maintenance event can take place:

(i) At least one repair team must be available.

(ii) The number of operating components and those on standby must be greater than k.

If any of the above conditions are not satisfied, the preventive maintenance event will be delayed. The component to be removed for preventive maintenance will be the one with the longest operating time since its last repair or maintenance.

2.2 Assumptions

(a) The component times to failure are independent and identically distributed random variables.

(b) The repair times are independent and identically distributed random variables.

(c) The component downtimes due to preventive maintenance are independent and identically distributed random variables.

(d) Repair and maintenance both restore the component to its original "as new" condition.

(e) Repair is performed on a "first-come, first-served" basis.

3. GPMS SOFTWARE PACKAGE

The GPMS software package is developed to model the performance of the system described above over a specified period. The GPMS requires the following inputs:-

(a) the number of components in the system,
(b) the number of repair teams,
(c) the operating policy,
(d) the preventive maintenance policy,
(e) the probability distribution of component times to failure,
(f) the probability distribution of repair times for failed components,
(g) the probability distribution of preventive maintenance downtimes,
(h) the number of simulation runs needed.

In the simulation, the program monitors the number of components in operation and the utilization of repair teams, taking into account events such as component failures, repairs and preventive maintenance. Initially, all the components in the system will be in "as new" condition. Since all the components have the same mean time to failure, it is possible that component failures may be lumped together rather than evenly spread out during the initial portion of the simulation. Simulation output statistics will not be gathered during this period. After some time, the simulation approaches the steady state, and statistics collected in the steady state are more realistic. A detailed discussion of initial conditions, simulation run times and statistics gathering is given in Appendix B.

The outputs of the simulation include:

(a) the probability that n components are available, n=0,1,2,...,N
(b) the mean number of components available
(c) the mean and standard deviation of the probability that the system provides full service

(d) the probability that n repair teams are busy, n=0,1,...,c
(e) the mean number of repair teams busy.

For a detailed discussion of the GPMS Software Package, refer to the User's Guide (Appendix A), Supplementary Information to User's Guide (Appendix B) and Programmer's Guide (Appendix C).

4. SAMPLE APPLICATION

The GPMS software package can be used to assess the availability of multi-component systems with specified operating and maintenance policies. Using these assessments, the most suitable operating and maintenance policies can be identified. As an example, consider a system of five identical components. The probability of this system failing to provide 100% service was assessed using the following data:

(a) Total number of repair teams = 1.

(b) Times to failure for each component follow an Erlang E_2 distribution with mean = 100 days.

(c) Repair times are exponentially distributed with mean = 5 days.

(d) Maintenance downtimes are exponentially distributed with mean = 2 days.

(e) Each of the following two operating policies was applied in turn in the assessment:

- All available components are placed in operation, with each of the components carrying an equal share of the load up to its capacity.
- Four components are operated to provide a capacity for exactly 100% service. The remaining component is placed on standby.

(f) Each of the following three maintenance policies was applied in turn in the assessment:

- No preventive maintenance.
- Group age maintenance policy where a component is removed for preventive maintenance 60 days after the completion of the last preventive maintenance event.
- Group block maintenance policy where a decision will be made every 60 days on whether a component should be removed for preventive maintenance.

Using the GPMS software package, the probability that the system fails to provide 100% service is determined for each of the specified operating and maintenance policies. The results are shown in Table 1. Note that for the system under consideration, preventive maintenance improves system

availability. In particular, the probability of system failing to provide 100% service is lowest if the system has the following operating and maintenance policies:

(i) Four components are operated to provide exactly 100% service, with each of these four components operated at full capacity. The remaining component is placed on standby.

(ii) At every 60 days, a decision will be made on whether a component should be removed for preventive maintenance.

TABLE 1

PROBABILITIES OF SYSTEM FAILING TO PROVIDE 100% SERVICE

Operating Policy \ Maintenance Policy	No Preventive Maintenance	Group Age Maintenance Policy	Group Block Maintenance Policy
All five components are placed in operation	0.0451	0.0400	0.0384
Not more than four components are placed in operation	0.0395	0.0268	0.0252

Note that finding (i) underlines the well-known fact that standby arrangement provides more reliability than parallel operation. However, there is some additional risk that when the component on standby is needed, it will not start or the switch will fail.

5. FURTHER WORK

Further research is planned to develop analytical models which will provide easily computable estimates of the reliability of the systems described in this report.

Approved:

for J.G. Cassan
Manager
Operations Research Department

Submitted:

D.F. Bartlett
Student
Reliability & Statistics Section

S.H. Sim
Engineer
Reliability & Statistics Section

REFERENCES

1. Bunday, B.D., and Scraton, R.E., "The G/M/r Machine Interference Model", European Journal of Operational Research 4(1980) 399-402.

2. Marie, R., "Calculating Equilibrium Probabilities for $\lambda(n)/C_k/1/N$ queues", Performance Evaluation Review 9 (1980) 117-125.

3. Maritas, D.G., and Xirokostas, D.A., "The $M/E_k/r$ Component Interference Model", European Journal of Operational Research 1(1977) 112-123.

4. Sim, S.H., "Reliability Model of a System of Identical Components with Non-Exponential Repair Times", Ontario Hydro Research Division Report No 85-31-K, 198

5. Stecke, K.E. and Aronson, J.E., "Review of Operator/Component Interference Models", Working Paper No 314, Graduate School of Business Administration, University of Michigan, 1982.

6. Stewart, W.J. and Marie, R., "A Numerical Solution for the $\lambda(n)/C_k/r/N$ queue", European Journal of Operational Research 5(1980) 56-58.

APPENDIX A

GPMS USER'S MANUAL

A1. Introduction

The Group Preventive Maintenance Simulation program is written in Micro-Soft FORTRAN IV and is designed to run on an IBM-compatible personal computer with an 8087 processor. If the 8087 is not included in the computer, the program will have to be recompiled with appropriate compiler information.

This simulation program consists of approximately 2200 lines of coding and is divided into two separately compiled sections. The first section gathers all input data. The second section is the main simulation program. Both sections of the program are divided up into many subroutines and functions. Input is done interactively through the keyboard and output may be produced on either the screen or a disk file.

A2. Requirements for Program Execution

(a) GPMS.EXE - This file is the executable program. It requires approximately 90K of memory space.

(b) SUFFICIENT DISK MEMORY

Unless you are sending the program output to the terminal screen, you will need sufficient disk memory to hold possibly a very large file. Without transaction logs, each simulation run requires almost 2K of disk memory. A program run consisting of 100 simulation runs requires approximately 170K disk memory. If transaction logs are produced, the amount of disk memory required for each simulation run will be considerably more, depending on the length of the simulation run, mean time to failure, preventive maintenance intervals and so on. Output on a hard-disk is recommended if transaction logs are requested.

A3. Execution Time

Execution time is dependent on a combination of the length of simulated operation in each run and other simulation parameters. The following is an example of time requirements needed for program execution:

Number of simulation runs = 10
Length of each simulation = 10000 days
Erlang-2 times to failure with mean = 100 days
Exponential repair times with mean = 5 days
Exponential preventive maintenance downtimes with mean = 2 days
Time to preventive maintenance $\geq$ 60 days
Total number of repair teams = 1
Total number of components = 5

Maximum allowable number of components operating = 5
Minimum number of operating components required = 4

Total program execution time = 10 minutes

A4. Operating the GPMS Program

The following notations are used in this appendix:-

PROMPT: This signifies that the following message will be displayed on the screen.

RESP: What you may enter in response to the prompt. A brief explanation is included whenever it is helpful to the user.

-> Signifies cursor location.

waiting... This generally appears when a message is displayed. No input is requested. However, the program will not continue until your press <ENTER>.

The steps in operating the program are:

(1) Change your directory to the directory which contains the file GPMS.EXE.

(2) Enter "GPMS" to run the program.

The first display to appear on the screen will contain some introductory information. Press <ENTER> to proceed.

(3) Select a method of data-entry.

The following information will appear.

You may choose one of two methods of input to enter data for this simulation package. The first method should be used for initial investigations of different strategies. The second method should be used for testing strategies with many different simulation run-time combinations or many different maintenance policies.

Input Method 1
This is more flexible than the second method by permitting different basic parameters such as the number of components and repair teams in a system. It also permits the option of producing a transaction log of the system as event occur.

Input Method 2
No transaction log may be produced. Input is less laborious than the first method.

PROMPT: ENTER A METHOD (1,2) ->

RESP: Enter either "1" or "2" as your desired input method.

(4) Enter output device number.
You will enter a device number concerning the destination of the simulation output.

PROMPT: Enter device number for output destination
0 - Screen
1 - Disk File for output on printer
* 2 - Disk File (no headings included)
(useful when requesting a transaction log)

ENTER (0,1,2) ->

RESP: Enter a 0, 1, (or 2 if using Input Method 1).

0 - All output is sent to the screen. If you press <CTRL> <P> simultaneously, your output will also be sent to your printer.

1,2 - This is a disk file, preferably a hard disk due to the generally large size of the output. 100 simulation runs without transaction logs require 170K disk space.

If the number entered is "1" or "2" then you will be prompted for a file name:-

PROMPT: Enter filename (eg. B:MYFILE>PRN) ->

RESP: A filename up to 21 characters in length may be entered.

* Note that only Input method 1 permits device number 2 to be chosen. It excludes all but the first set of headings in the transaction log. This is useful for saving disk space and simplifying data modifications so that data may be read by another program (eg. Lotus 123).

(5) Enter computational units.
An example of time to component failure may be 10 hours, 10 days, etc. Up to 10 characters in length may be entered.

PROMPT: Enter units of computation (Hours, Days, ...) ->

RESP: Enter: Hours
Days ... etc up to 10 characters

(6) Enter random number generation seeds.

PROMPT: You must enter 3 random number generation seeds of different values. These must range between 1 and 29999.

RESP:
Enter first seed ->
Enter second seed ->
Enter third seed ->

These 3 seeds are important for good random number generation. Try to enter prime numbers as seeds, for they will generally provide the best results.

Now the user's guide is broken up into two parts, depending upon which method of input is used.

If using "Input Method 1" proceed to (7.1)

If using "Input Method 2" proceed to (7.2)

(7.1) User's Guide for Input Method 1:

(A)

PROMPT: Do you need a transaction log in your output? Y/N
(default is No) ->

RESP: Enter Y for yes or press <ENTER> for No as the default

The transaction log is a sequential listing of events occurring in the simulation. It produces a log of each event occurring for the duration of the simulation run. For further information, see the section on Simulation Output concerning transaction logs.

(B)

PROMPT: Enter starting time for collection of statistics in DAYS
(maximum value which can be entered is 30000) ->

RESP: Now you must enter the starting time for gathering statistics. At the start of each simulation run, each component is in "as new" condition. This does not represent a realistic system. We say that the system is not in steady state. A certain period of time will elapse before the system reaches steady state. It is important that the collection of statistics does not start until this steady state is reached. For further information, see the section concerning Simulation Run Times and Statistics Gathering.

(C)

PROMPT: Enter total simulation run time in DAYS
(maximum value which can be entered is 30000) ->

RESP: Enter total run time for each simulation. It must be greater than the starting time for collection of statistics. For further information, see the section concerning Simulation Run Times and Statistics Gathering.

(D)

PROMPT: Enter total number of repair teams (Max = 100) ->

RESP: Total number of repair teams represents the maximum number of components which may undergo repair/maintenance. If more components are in an inoperable state than repair teams available, the components are placed in a queue to await repair. Components in the queue undergo repair on a "first-come, first-served" basis.

(E)

PROMPT: Enter total number of components in the system.
Maximum # = 1099 (100 Max in operation, 999 Max on standby)

RESP: Enter the total number of components in the system.

(F)

PROMPT: Enter maximum number of components to run simultaneously
Maximum # = lesser of 100 or Total # components ->

RESP: Enter the maximum number of components which may be operated at one time. The number entered must be the lesser of either 100 components or the total number of components in the system.

(G)

PROMPT: Enter minimum number of components to run simultaneously
Minimum # $\leq$ Min (100, Maximum # components) ->

RESP: Enter the minimum number of components which must be operated for 100% service. Note that preventive maintenance will only take place when the number of components in operation, plus the number of components on standby, is greater than the required minimum number of components needed for full service.

(H)

PROMPT:

```
1 -No preventive maintenance policy
2 -Group Age preventive maintenance policy
3 -Group Block preventive maintenance policy
4 -Individual Age preventive maintenance policy

Enter a preventive maintenance policy (1,2,3,4)   ->
```

RESP: Enter one of the four preventive maintenance policies. For further information, see the section titled "Preventive Maintenance Policies".

(I) A series of two or three prompts (depending on preventive maintenance policy) will request distribution identification numbers. These will determine the random variates for times to failure, repair times for failed components and preventive maintenance downtimes.

PROMPT:

IDENTIFICATION #	DISTRIBUTION
1	EXPONENTIAL
2	ERLANG (GAMMA)
3	WEIBULL
4	LOGNORMAL
5	UNIFORM

```
Enter ID. # for time to failure distribution   ->
Enter Mean ->
```

RESP: Enter one of the above distribution identification numbers. You will then be prompted for the necessary parameters needed by the distribution. These prompts will be repeated for repair times of failed components and, if applicable, preventive maintenance downtimes.

(J)

PROMPT: For accurate results, you must repeat the simulation run many times. This will enable you to obtain mean and standard deviation of the probability that the system fails to provide 100% service. You are allowed up to 200 simulation runs for each set of input data.

```
Enter how many runs you would like:
Maximum = 200, default = 1   ->
```

RESP: Due to the sensitivity of this software package to different seeds, it is necessary to perform many simulation runs for each

set of input data. A mean and a standard deviation is then produced. For most strategies, 50 to 100 runs may be sufficient to produce accurate results.

(K)

PROMPT: If you have made any errors while entering data into this simulation run, you may re-enter the data by typing YES to the following prompt.

Enter YES or press <ENTER> for NO ->

RESP: If you have made any mistakes while entering data, you may start over by entering YES to the prompt. Steps B to K will be repeated.

(L)

PROMPT: Press <ENTER> to enter another simulation run, or type "S" to stop data-entry and start simulation: ->

RESP: Enter "S" when you don't want to enter any more data. If you press <ENTER>, steps B to L will be repeated. If you have entered 100 simulation runs, simulation will start automatically.

(7.2) User's Guide for Input Method 2:

(A) The following message will appear on the screen:-

"Enter starting time for collection of statistics and ending time for total length of the simulation run. Several different sets of simulation times may be entered. (Maximum is 100)"

You may repeat steps B and C up to 100 times (until you enter No to the prompt in step D). Entering several different times may be useful for establishing proper times for both collection of statistics and total simulation run time.

(B)

PROMPT: Enter starting time for collection of statistics in DAYS
(maximum value which can be entered is 30000) ->

RESP: Now you must enter the starting time for gathering statistics. At the start of each simulation run, each component is in "as new" condition. This does not represent a realistic system. We say that the system is not in steady state. A certain period of time must elapse before the system reaches a steady state. It is important that the collection of statistics does not start until

this steady state is achieved. For further information, see the section concerning Simulation Run Times and Statistics Gathering.

(C)

PROMPT: Enter total simulation run time in DAYS
(maximum value which can be entered is 30000) ->

REPS: Enter total run time for each simulation. It must be greater than the starting time for collection of statistics. For further information see the section concerning Simulation Run Times and Statistics Gathering.

(D)

PROMPT: Do you want to enter another set of times? Y/N
(default is Yes) ->

RESP: Press <ENTER> to enter another set of times. Enter No to stop entering times and proceed to step E. You may repeat steps B and C up to 100 times.

(E)

PROMPT: Enter total number of repair teams (Max = 100) ->

RESP: Total number of repair teams represents the maximum number of components which may undergo repair/maintenance simultaneously. If more components are in an inoperable state than repair teams available, the components are placed in a queue to await repair. Components in the queue undergo repair on a "first-come, first-serve" basis.

(F)

PROMPT: Enter total number of components available
Maximum # = 1099 (100 Max in operation, 999 Max on standby)

RESP: Enter the total number of components in the system. Maximum is 1099, of which a maximum of 100 Components may be placed in operation and a maximum of 999 components may be placed on standby at any one time.

(G)

PROMPT: Enter maximum number of components to run simultaneously
Maximum # $\leq$ lesser of 100 or Total # components ->

RESP: Enter the maximum number of components which may be operated at one time. The number entered must be the lesser of either 100 components or the total number of components in the system.

(H)

PROMPT: Enter minimum number of components to run simultaneously
Minimum # ≤ Min(100,Maximum # components) ->

RESP: Enter the minimum number of components which must be operated for 100% service. Note that preventive maintenance will only take place when the number of components in operation, plus the number of components on standby, is greater than the required minimum number of components needed for full service.

(I)

PROMPT: Enter total number of repair teams (Max = 100) ->

RESP: Total number of repair teams represents the maximum number of components which may be under repair/maintenance simultaneously. If more components are in the inoperable state than repair teams available, the components are placed in a queue to await service. Components in the queue are served on "First In - First Out" basis.

(J)

The following message will appear on the screen:-

Now you must enter which preventive maintenance policy is desired, along with means and any other needed parameters for:

1. times to failure,
2. repair times for failed components,
3. downtimes due to preventive maintenance.

You may enter up to XX strategies

waiting ...

The above message introduces you to the next data-entry step. Press <ENTER> to continue with data entry. A number is displayed in place of XX. this tells you how many strategies may be entered. XX is computed by the following formula (any resulting remainder dropped):-

XX = 100 / (# of different sets of run times entered)

eg. If 2 different sets of run times were entered then you may enter up to 50 strategies. If 3 sets of run times were entered then you may enter up to 33 strategies.

(K)

PROMPT:

1 - No preventive maintenance policy
2 - Group Age preventive maintenance policy
3 - Group Block preventive maintenance policy
4 - Individual Age preventive maintenance policy

Enter a preventive maintenance policy (1,2,3,4) ->

RESP: Enter one of the four preventive maintenance policies. For further information, see Appendix B2.

If more than one strategy has been entered and the last strategy had the same type of preventive maintenance policy, then you will be prompted with the following message:-

PROMPT: Enter Yes to use the same failure, repair, and preventive maintenance parameters as was entered previously (default in No) ->

RESP: Enter Yes to repeat the same parameters used in the last strategy with this one.

(L)

A series of two or three prompts (depending on preventive maintenance policy) will request distribution identification numbers. These will determine the random variates for times to failure, repair times for failed components and downtimes due to preventive maintenance.

PROMPT:

IDENTIFICATION #	DISTRIBUTION
1	EXPONENTIAL
2	ERLANG (GAMMA)
3	WEIBULL
4	LOGNORMAL
5	UNIFORM

Enter ID. # for time to failure distribution ->
Enter Mean ->

RESP: Enter one of the above distribution identification numbers. You will then be prompted for the necessary parameters needed by the distribution. These prompts will be repeated for the repair times of failed components, and if applicable, preventive maintenance downtimes.

(M)

PROMPT: For accurate results, you must repeat the simulation run many times. This will enable you to obtain mean and standard deviation of the probability that the system fails to provide 100% service. You are allowed up to 200 simulation runs for each set of input data. Note that each run will require about 1 minute of computer time.

Enter how many runs you would like:
Maximum = 200, default = 1 ->

RESP: Due to the sensitivity of this software package to different seeds, it is necessary to perform many simulation runs for each set of input data. A mean and a standard deviation is then produced. For most strategies, 50 to 100 runs may be sufficient to produce accurate results.

(N)

PROMPT: You have entered XX strategies
Enter another strategy? Y/N (default is Y) ->

RESP: Enter Yes if you would like to enter another preventive maintenance policy and parameters for that policy (steps J to M will be repeated).

A5. Sending Output to a Printer Connected to PC

Output may be sent to your printer from a disk file, or displayed on the screen (while the program is executing).

To send data from a disk file to the printer enter the following:-

```
C> PRINT B:MYFILE.PRN
```

If you are sending your output to the screen and would also like to have a printout, then (while the program is running) simultaneously press the following keys to activate the printer:-

<CTRL> <P>

When the execution is completed, or you have obtained all the information you want, repeat the above process to turn the printer off.

APPENDIX B

SUPPLEMENTARY INFORMATION TO USER'S GUIDE

B1. Simulation Run Times and Statistics Collection

Accurate results may not be obtained if proper times are not set for starting statistics collection. Initially, all the components in the system will be in "as new" condition. Since all the components have the same mean time to failure, it is possible that component failures may be lumped together rather than evenly spread out during the initial portion of the simulation. Eventually, the system will be in steady state, and the results will be more realistic. Steady state will exist only if the total failure rate of components (N/MTTF) is less than the total repair rate (c/MTTR). "Throw-away" is referred to the initial period of simulation run when the system is not in steady state. Statistics will be collected when the system is in steady state. The software package will prompt the user for a starting time for the collection of statistics. This is equivalent to the throw-away time.

If one is just investigating a strategy and do not want to go through the effort of determining proper throw-away, a simple rule which should yield relatively accurate results is to multiply the component's mean time to failure (MTTF), plus its mean time to repair (MTTR), by 40. Use the result as the throw-away time. This pattern has been observed during testing of the simulation package; the first with MTTF = 100 days, MTTR = 5 days; the second with MTTF = 50 days, MTTR = 5 days. In both testing procedures only 5 components were in the system, all of which were in operation. Note that making the throw-away time longer will not worsen the accuracy of the results, but it may not improve them significantly either, and the runs become more expensive.

One may wish to use Input Method 2 to determine the throw-away time accurately. Enter a variety of total simulation run times, each time incrementing the total run time by a set value. For example:-

Starting Time for Collection of Statistics	Total Simulation Run Time
0 days	1000 days
.	1500 days
.	2000 days
.	2500 days
.	.
.	.
0 days	10000 days

Then, while in Input Method 2, enter your parameters for the strategy. Test the strategy with 25 runs (each set of simulation times will produce a mean of 25 runs). Making a plot of the average number of the components in operation is the easiest way to determine the throw-away. Figure B1 is an example of

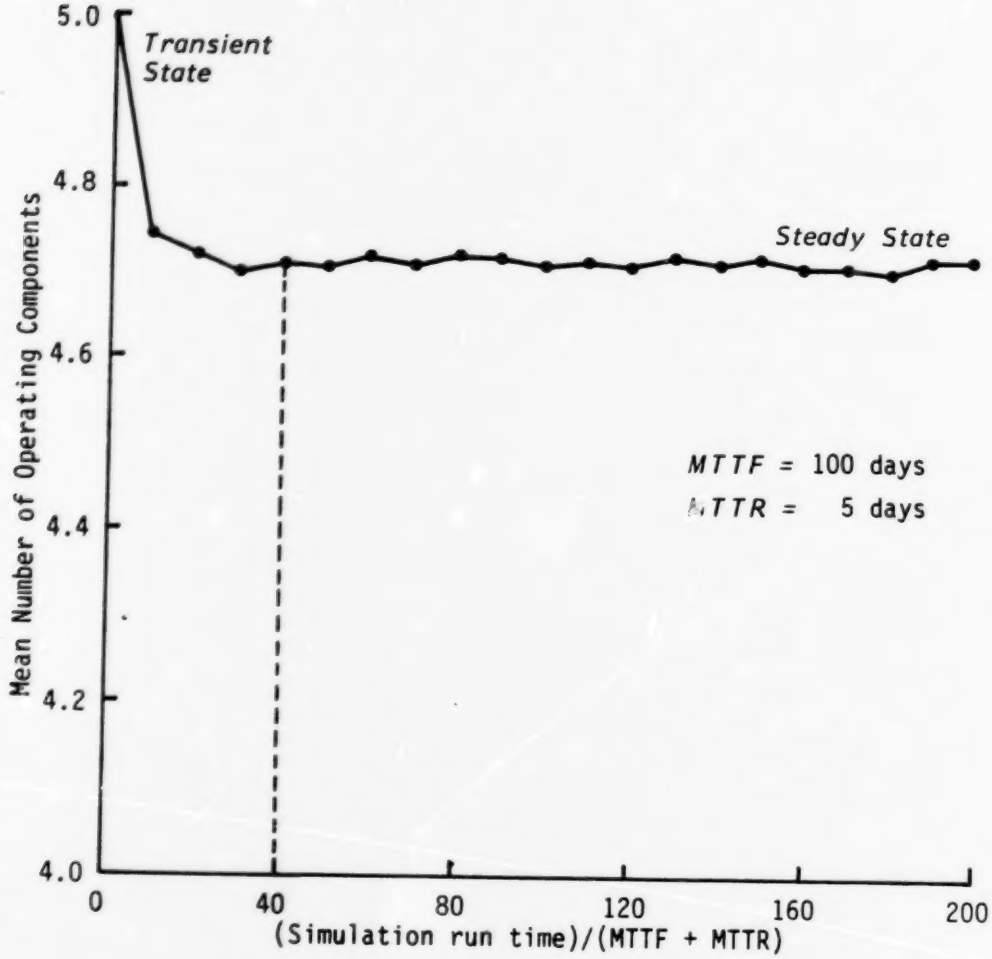

FIGURE B1

THE EFFECT OF SIMULATION RUN TIME ON THE COLLECTION OF STATISTICS FOR ESTIMATING THE MEAN NUMBER OF OPERATING COMPONENTS

the plot for a system of 5 components (each of which has Erlang E_2 times to failure) and no preventive maintenance is allowed. The starting time for statistics collection is located where the graph levels off and appears to be relatively horizontal and constant. This is the approximate time when steady state is attained. In Figure B1, it appears that the system approaches the steady state for a simulation run length exceeding 40 times the sum of mean time to failure and mean repair time.

The simulation run time may be obtained as follows: "Simply add the mean time to failure to the mean repair time and multiply the sum by 50. Add this sum to the throw-away time." This should allow a sufficient number of breakdown and repair cycles to occur for producing accurate statistics.

B2. Preventive Maintenance Policies

In this simulation model, preventive maintenance is defined as an activity undertaken at fixed but judiciously chosen intervals while the component is satisfactorily working, to restore it to "as new" condition. Preventive maintenance is performed on the component which has been in operation the longest.

Each simulation run requires one of the following.

1) No preventive maintenance
2) Group Age preventive maintenance
3) Group Block preventive maintenance
4) Individual Age preventive maintenance

The necessary conditions under which preventive maintenance events can take place are: -

(a) At least one repair team must be available.

(b) The number of components in operation, plus the number of components on standby, must be greater than the minimum number of components required for full service.

An explanation of each policy is given as follows:-

1) No preventive maintenance refers to a policy of only performing repairs on failed components.

2) Group Age preventive maintenance refers to a policy of performing preventive maintenance only after a fixed elapsed time since completion of the last preventive maintenance event. If the conditions are not favorable, preventive maintenance will be delayed until conditions become acceptable. For example: -

Preventive maintenance time interval = 50 days

SIMULATION TIME	EVENT
1000 days	Preventive maintenance has just been completed on a component.
1050 days	50 days has elapsed since completion of last preventive maintenance. Conditions are favorable for a component to be removed for preventive maintenance.
1055 days	After 5 days, preventive maintenance has been completed.
1105 days	50 days has elapsed since completion of the last preventive maintenance. Conditions are not favorable for preventive maintenance.
1115 days	This represents the first time (since time 1105 days) conditions have become favorable for preventive maintenance. A component is removed for preventive maintenance.
1120 days	After 5 days, preventive maintenance has been completed.
1170 days	50 days has elapsed since completion of last preventive maintenance.

3) Group Block preventive maintenance refers to a policy of performing preventive maintenance at fixed (scheduled) intervals. It is very similar to Group Age preventive maintenance except that there is no minimum time elapsed between two consecutive preventive maintenance events. If conditions are not favorable, preventive maintenance will be delayed, but scheduled times of future preventive maintenance actions do not change. If conditions do not become favorable for preventive maintenance by the time the next preventive maintenance epoch is reached, preventive maintenance is canceled. For example:-

Preventive maintenance time interval = 50 days

SIMULATION TIME	EVENT
1000 days	50 days has elapsed since the last scheduled preventive maintenance. Conditions are favorable for a component to be removed for preventive maintenance.
1005 days	After 5 days, preventive maintenance has been completed.
1050 days	50 days has elapsed since the last scheduled preventive maintenance. Conditions are not favorable for preventive maintenance.
1075 days	Conditions become favorable for preventive maintenance. A component is removed for preventive maintenance.
1080 days	After 5 days, preventive maintenance is completed.
1100 days	50 days has elapsed since the last scheduled preventive maintenance. Conditions are not favorable for preventive maintenance.

1150 days	Another 50 days has elapsed since the last scheduled preventive maintenance. The past scheduled preventive maintenance (time 1100 days) is canceled. 1160 days Conditions become favorable for preventive maintenance .

4) Individual Age preventive maintenance refers to a policy of performing preventive maintenance on each component after it has operated for a fixed time period. No practical system of scheduling can be attained for preventive maintenance with this policy. For example:-

Preventive maintenance time interval = 50 days

SIMULATION TIME	EVENT
1000 days	Component number 3 has been in operation for the past 50 days. Conditions are favorable for preventive maintenance. Component number 3 is removed.
1005 days	After 5 days, preventive maintenance has been completed on component number 3.
1015 days	Component number 1 has been in operation for the past 50 days. Conditions are not favorable for preventive maintenance.
1020 days	Conditions become favorable for preventive maintenance. Component number 1 is removed for preventive maintenance.
1025 days	After 5 days, preventive maintenance has been completed on component number 1.
1035 days	Component number 2 has been in operation for the past 50 days. Conditions are not favorable for preventive maintenance.
1040 days	Component number 5 has been in operation for the past 50 days. Conditions are not favorable for preventive maintenance.
1050 days	Conditions for preventive maintenance have become favorable. Since component number 2 has been in operation the longest (65 days vs. 60 days), it is removed for preventive maintenance.

B3. Random Variate Generation

A uniform random number generator called RANDU is used to generate other random variate distribution. RANDU generates numbers that are uniformly distributed between 0 and 1.

The following distributions and their parameters are included in this software package:

Distribution Id #	Distribution	Parameters
1	Exponential	Mean
2	Erlang (Gamma)	Mean, R
3	Weibull	Mean, Shape
4	Lognormal	Mean, Std. Dev.
5	Uniform	Mean

The following is a discussion of the use of each of these distributions:-

(1) GPMS routine name - EXPON

Purpose - Exponential random variate generation

Arguments - X - Output parameter containing the value generated by EXPON.
MEAN - Input parameter containing the mean.

Usage - X = EXPON (MEAN)

Precision - Double

Required GPMS routines - RANDU

Algorithm:

EXPON generates a pseudo-random variate from the Exponential distribution, with density function

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \lambda > 0,\ x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and distribution function

$$F(x) = 1 - e^{-\lambda x} \qquad x \geq 0$$

By inversion, an Exponential variate may be generated as

EXPON = mean * [-ln(u)]

where u is a pseudo-random number from a uniform (0,1) distribution.

(2) GPMS routine name - ERLANG

Purpose - Erlang (Gamma) random variate generation

Arguments - X - Output parameter containing the value generated by ERLANG.
MEAN - Input parameter containing the mean.
R - Input parameter. It specifies the degree of variability of the generated time relative to the mean. R be an integer > 0, However it is stored in memory as Double Precision.

Usage - X = ERLANG (MEAN,R)

Precision - Double

Required GPMS routines - EXPON

Algorithm:

ERLANG generates a pseudo-random variate from the Erlang (Gamma) distribution, with density function

$$f(x) = \begin{cases} \dfrac{r\lambda(r\lambda x)^{r-1} e^{-r\lambda x}}{(r-1)!} & x \geq 0, \quad \lambda > 0, \; r=1,2,.. \\ 0 & \text{otherwise} \end{cases}$$

and distribution function

$$F(x) = 1 - \sum_{j=0}^{r-1} \frac{(r\lambda x)^j}{j!} e^{-r\lambda x}$$

An Erlang (Gamma) variate may be generated as a sum of r exponential variates, each of mean $1/(r\lambda)$.

(3) GPMS routine name - WEIBULL

Purpose - Weibull random variate generation

Arguments - X - Output parameter containing the value generated by WEIBULL.
MEAN - Input parameter containing the mean.
SHAPE - Input parameter containing the shape for the desired Weibull function. Shape must be greater than 0.

Usage - X = WEIBULL (MEAN,SHAPE)

Precision - Double

Required GPMS routines - RANDU

Algorithm:

WEIBULL generates a pseudo-random variate from the Weibull distribution, with density function

$$f(x) = \begin{cases} sx^{s-1} \exp(-x^s) & x \geq 0, \quad s > 0 \\ 0 & \text{elsewhere} \end{cases}$$

and distribution function

$$F(x) = 1 - \exp(-x^s)$$

By inversion, a Weibull variate may be generated as

$$\text{WEIBULL} = [-\ln(u)]^{1/s}$$

where s is the shape and u is a pseudo-random number from a uniform (0,1) distribution.

(4) GPMS routine name - LOGNORM

Purpose - Log-normal random variate generation

Arguments - MEAN - Input parameter containing the mean.
STD - Input parameter containing the standard deviation of the desired Log-normal function. The standard deviation must be greater than 0.

Time - Output parameter containing the variate generated.

Usage	-	CALL LOGNORM (MEAN,STD,TIME)
Precision	-	Double
Required GPMS routines	-	RANDU, EXPON

Algorithm:

LOGNORM generates a pseudo-random variates.
Variates are generated from the log-normal distribution having density function

$$f(x) = \begin{cases} \frac{1}{\sqrt{2\pi}\,\sigma x} \exp[-(\ln x-\mu)^2/(2\sigma^2)] & x \geq 0,\ \sigma > 0 \\ 0 & \text{elsewhere} \end{cases}$$

Normal (0,1) pseudo random variates u are transformed to log normal deviates x, as

$$X = \exp(\mu + \sigma u)$$

(5) The UNIFORM distribution is not calculated in a separate routine as are the others. It is derived in a simple formula which makes a call to RANDU. Its format is as follows:

$$X = \mu u + 0.5$$

where u is a pseudo-random number from a uniform (0,1) distribution (RANDU).

B4. Simulation Output

B4.1 Statistical Output of Simulation Results:

The output consists of a summary of the system parameters and statistical results of the simulation. This output is produced for each and every simulation run. If a transaction log was also requested for a simulation run, it will be produced immediately before the statistical output. An example of the statistical output of simulation results is as follows:-

```
SIMULATION RUN NUMBER  1

PREVENTIVE MAINTENANCE POLICY - 2
TOTAL NUMBER OF REPAIR TEAMS AVAILABLE            -    1
TOTAL NUMBER OF COMPONENTS AVAILABLE              -    5
MAX NO. OF COMPONENTS TO OPERATE                  -    5
MIN NO. OF COMPONENTS REQUIRED FOR FULL SERVICE   -    4
TIME TO PREVENTIVE MAINTENANCE OCCURS             ≥ 20.0 DAYS


FAILURE TIMES SIMULATED WITH THE ERLANG DISTRIBUTION
   MEAN :      100.00
   R :           2
REPAIR TIMES SIMULATED WITH THE EXPONENTIAL DISTRIBUTION
   MEAN :        5.00
PREVENTIVE MAINTENANCE DOWN TIME SIMULATED WITH EXPONENTIAL
   MEAN :        2.00

STARTING TIME OF GATHERING STATISTICS -  5000.0 DAYS
TOTAL SIMULATION RUN TIME -             15000.0 DAYS

FIRST RANDOM NUMBER GENERATION SEED     :   7651.0
SECOND RANDOM NUMBER GENERATION SEED    :   9811.0
THIRD RANDOM NUMBER GENERATION SEED     :    453.0

RESULTS OF SIMULATION

AVAILABILITY OF COMPONENTS:
PROB (  5 COMPONENTS OPERATING  ) -   0.7551
PROB (  4 COMPONENTS OPERATING  ) -   0.2209
PROB (  3 COMPONENTS OPERATING  ) -   0.0237
PROB (  2 COMPONENTS OPERATING  ) -   0.0002
PROB (  1 COMPONENTS OPERATING  ) -   0.0000
PROB (  0 COMPONENTS OPERATING  ) -   0.0000
AVAILABILITY OF REPAIR TEAMS:
PROB (  1 REPAIR TEAMS BUSY  )    -   0.2449
PROB (  0 REPAIR TEAMS BUSY  )    -   0.7551

PROB ( LESS THAN   4 COMPONENTS OPERATING ) -    0.0239
AVERAGE NUMBER OF COMPONENTS OPERATING -         4.73
AVERAGE NUMBER OF REPAIR TEAMS BUSY -            0.2449
```

B4.2 Transaction Logs:

The transaction log is an optional addition to the regular statistical output produced for every simulation run. Transaction logs may be produced using Input Method 1.

The transaction log is a sequential listing of events occurring in a simulated operation. It produces a log of all the events taking place for the duration of a simulation run. The following is an example of a transaction log: -

TABLE B4: TRANSACTION LOGS

CLOCK	BR	RP	PM	COMPONENT	ONL	OFFL	STANDBY	BUSY
68.11	1	0	0	5	4	1	0	1
75.04	0	1	0	5	5	0	0	0
75.04	0	0	1	3	4	1	0	1
75.31	0	1	0	3	5	0	0	1

CLOCK - Time at which an event takes place (days)
BR - Event taking place was a component failure
RP - Event taking place was a completion of repair/ maintenance
PM - Event taking place was a component removed for preventive maintenance.
COMPONENT - Component number upon which an event took place
ONL - Current number of components in operation.
OFFL - Current number of components not in operable condition.
STANDBY - Number of components in 'as new' condition, but are not in operation.
BUSY - Number of repair teams busy.

The above table shows that at time 68.11 (days) component number 5 has failed. At time 75.04, repair on component 5 was completed and it was put back in operation (ONL). At the same time, conditions became favorable for component 3 to be removed for preventive maintenance. Completion of preventive maintenance occurred at time 75.31, and the component was put back in operation.

APPENDIX C

PROGRAMMER'S GUIDE

C1. Functional Descriptions

The program logic consists of the following steps:

STEP 1:

Get all input data. The input routine will set the number of simulation runs to be executed and the parameters used in each run.

STEP 2:

Initialize computational variables. These variables include the clock time, statistics recording variables, and so on. All components will be set to "as new" condition and future component failure times will be determined for those initially put in operation. If the total number of components in the system is greater than the maximum number of components allowed to operate, the remaining components will be put on standby in "as new" condition.

STEP 3:

The simulation time will be advanced to the next logical event to take place. This event may be a completion of repair/maintenance (with the exception of the first event of the simulation), a failure or preventive maintenance. However, preventive maintenance will occur only if a repair team is available and the number of components in operable condition is greater than the minimum number of components required for full service.

Included in this step is statistics collection. Statistics are recorded once the simulation time has surpassed the starting time for collection of statistics.

STEP 4:

This step is performed if the event determined by step 3 is a component failure. The failed component will be taken off-line. If another component is on standby, it will be placed in operation (on-line). The component taken from standby will have a time to failure assigned to it. If a repair team is available, the failed component will be assigned a time for completion of repair. Otherwise the component will be placed at the end of a queue to await repair.

STEP 5:

This step is performed if the event determined by step 3 is the completion of repair/maintenance. The component on which repair/maintenance has just completed will either be placed in operation or on standby. If the number of components in operation is less than the maximum allowed by the strategy, the component will be placed in operation and a new time to failure will be assigned to it. Otherwise, the component will be placed on standby.

STEP 6:

If neither a failure nor a repair event occurs in step 3, then the event to take place must be preventive maintenance. The component to be maintained is taken off-line and a time to completion of preventive maintenance is assigned to it. As required by the specified operating policy, a standby component may be placed on-line. A time to failure is assigned to this component.

STEP 7:

If a transaction log is wanted for a specific simulation run, then a dump of the state of the system is produced for each and every event taking place during the simulation. The dump includes the simulation time for each event; whether the event was a failure, repair, or preventive maintenance; the component (as represented by identification number) being acted upon; and the status of the system. The status of the system shows how many components are in operation; how many are not in operational order; the number of components on standby; the number of repair teams busy.

STEP 8:

If the simulation time is less than the required length of simulation, steps 3 to 7 will be repeated.

STEP 9:

After the simulation has been completed for the current run, the results will be produced on the output device. This output includes all the parameters required for the simulation, such as the total number of repair teams, total number of components, random number generation seeds, etc ... The statistics produced will show the probability distribution of the number of components operating simultaneously, the probability distribution of the number of repair teams busy, the mean and standard deviation of the probability that the system fails to provide 100% service, the average number of components operating, and the average number of repair teams busy.

STEP 10:

Repeat steps 2 to 9 until all simulation runs entered by the user are simulated.

C2. GPMS Program Segments

Internal functions and subroutines used within the main program GPMS.FOR:-

ADVANCE
FAILURE
ERLANG
EXPON
GATHER
GIVEDIST
INIT
LOGNORM
PMROUT
POPQ
PUSHQ
RANDU
REPARR
RESULTS
TIMES
THROW
TXNLOG
WEIBULL
WRITEPARAMS

External subroutines called by GPMS.FOR: -

CLEAR
INPUT

Internal functions and subroutines used within the external subroutine INPUT.FOR:-

ASKPARAMS
ICHAR2
INDIST
INPM
INSEED
INTRO
LISTDISTS
NUMRUN
RCHAR8
TESTS

External subroutines called by GPMS.FOR: -

CLEAR

C3. Description of Routines and Functions

ADVANCE - Advances the simulation time clock to the next logical event to take place. It determines whether the next event is a failure, completion of repair/maintenance, or preventive maintenance. It determines the number of the component to which an event is to take place. Logical flags determine the event.

ASKPARAMS - Prompts the user to enter parameters for the random variate generation. The distribution identification number must be known before entry to this routine. The first parameter passed back from this routine is always the mean. The second parameter is dependent upon the distribution specified. It may be a shape, a standard deviation, etc...

TABLE C1 - Available Distributions

Distribution Id #	Distribution	Parameters
1	Exponential	Mean
2	Erlang	Mean, R
3	Weibull	Mean, Shape
4	Lognormal	Mean, Std. Dev.
5	Uniform	Mean

CLEAR - External Assembly Language subroutine to clear the screen and place the cursor in the upper left hand corner.

ERLANG - Erlang variate generator.

EXPON - Exponential variate generator.

FAILURE - This routine performs the necessary steps following a component failure. It takes the component off-line and begins repair on it. If all the repair teams are busy, it places the component at the end of the repair queue to wait for repair. If available, a standby, component is automatically placed in operation (on-line).

GIVEDIST - Returns the name of the distribution from its corresponding identification number.

ICHAR2 - Converts a character array into an integer*2 value.
Input: character*1 array of dimension 10.
Output: (a) integer*2 value
(b) logical*2 error flag raised when an invalid number is entered.

INDIST - Prompts the user to specify a probability distribution function and its parameters for:-
(1) times to failure
(2) repair times for failed components
(3) downtimes due to preventive maintenance

INIT - Initializes all variables before each simulation run. This includes setting the clock time to zero and determining times to failure for the components to be placed in operation (on-line).

INPM - Prompts the user to specify a preventive maintenance policy and time interval for preventive maintenance.

INPUT - External Micro-Soft FORTRAN subroutine responsible for collecting all input data. Input Method 1 utilizes this routine for control of all input. Input Method 2 utilizes the TESTS routine for a majority of input control.

INSEED - Prompts the user to assign three initial seeds for random number generation.

INTRO - Displays introductory information about the package and prompts the user for one of two methods of entering data.

LISTDISTS - Display the probability distributions available for application in the simulation. (See Table C1).

LOGNORM - Lognormal variate generator.

NUMRUN - Prompts the user to specify the number of times a strategy is to be tested (with different seeds).

PMROUT - Performs the necessary steps for preventive maintenance. A component is taken off-line and, if applicable, a standby component is placed in operation (on-line).

POPQ - Removes a component from the repair queue for repair to start.

PUSHQ - Places a failed component at the end of the repair queue. Components are placed in the queue when all repair teams are busy.

RANDU - Uniform random number generator. Generates Pseudo-random numbers uniformly distributed between 0 and 1.

RCHAR8 - Converts a character array into a real number.
Input: character*1 array of dimension 10.
Output: (a) real*8 value
(b) logical*2 error flag raised when an invalid number is entered.

REPARR - Restores a component from repair/ maintenance to operable condition. If the number of components in operation is less than the maximum number allowed to operate, the component is put in operation (on-line). Otherwise the component is put on standby. Repair will begin on the component at the top of the repair queue (if there is any).

RESULTS - Produces, as output, the results of each simulation run. This includes a summary of the input parameters and the statistical results.

TESTS - Virtually the same as the INPUT routine, except that data-entry is arranged differently. This routine is specialized for testing strategies. Input Method 2 utilizes this routine.

THROW - Throws away the first 20000 random numbers produced by the uniform random number generator (RANDU). If the user enters some bad seeds, they will have a chance to correct (normalize) themselves after 20000 calls to RANDU.

TIMES - Generates random variates from distributions such as Erlang distribution, Exponential distribution, and so on. These random variates are needed for times to failure, repair times, and downtimes due to preventive maintenance.

TXNLOG - Produces a transaction log on the output device. This log includes the simulation time that an event takes place, the type of event which has taken place, and the current state of the system.

WEIBULL - Weibull variate generator.

WRITEPARAMS - Prints, on the output device, the parameters for a particular distribution which may be applied to times to failure, repair times, and downtimes due to preventive maintenance. Parameters are shown in Table C1.

C4. Variables

The following variables are used throughout both the main program (GPMS.FOR) and the input program (INPUT.FOR).

AMEANS - An array containing the means of several simulation runs so that an overall mean and standard deviation can be attained.

BUSY - Number of repair teams working simultaneously.

CLOCK - Current simulation run time (in units specified by the user - hours, days, ...).

CNTBSY - Recorded statistics of the number of repair teams busy.

CNTMCH - Recorded statistics of the number of components operating simultaneously.

DEVICE - Output device number. 0 for screen, 1 for disk file formatted for printer, 2 for disk file with only one set of headings printed in the log.

FDIST - Contains time to failure parameters. These are unique for every simulation run, and are used for random variate generation.

FRONT - Pointer to the front of the repair queue. It holds component's numbers when all repair teams are busy.

LINECNT - Number of lines that has been produced on the output device. Used for determining head intervals in the transaction logs.

MACH - The current status of each component during simulation.

MACH(I,1)
- The clock time of next failure if component I is operating.
- 32767 (Maximum allowable single precision integer) if component I is not operating (i.e. in failed state or standby).

MACH(I,2)
- The clock time of repair completion if component I is undergoing repair.
- -1 if component I is in operable or operating state.
- 32767 (Maximum) if component I is in repair queue, waiting for repair.

MACH(I,3)
- The clock time of the last completion of maintenance or repair if component I is operating.
- -1 if component I is on standby.
- 32767 (Maximum) if component I is in failed state.

PMMODE - Contains preventive maintenance policy number and time to preventive maintenance for each simulation run (where applicable).

MAXMCH - Maximum number of components allowed to operate simultaneously in each particular simulation run.

MAXIMUM - Maximum allowable single-precision integer. It is used for logical manipulations.

MDIST - Contains the identification numbers of the various distributions available. These distributions are used for random variate generation as defined by:

MDIST (I,J) = 1 for Exponential distribution
= 2 for Erlang distribution
= 3 for Weibull distribution
= 4 for Lognormal distribution
= 5 for Uniform distribution
where I = simulation run number
J = 1 for time to failure
= 2 for repair time
= 3 for downtime due to preventive maintenance

MINMCH - Minimum number of components to operate simultaneously in each particular simulation run.

NSTRAT - Total number of simulation runs entered by the input routine and performed by the simulation.

OFFL - Number of components off-line (not in operational order due to repair/maintenance).

ONL - Number of components on-line (in operation).

PMDIST - Contains preventive maintenance downtime parameters. These are used for random variate generation.

QUEUE - Repair queue which holds component identification numbers for components awaiting repair

RDIST - Contains parameters for time to completion of repair for failed components. These are used for random variate generation.

REAR - Pointer to rear of the repair queue which holds components when all repair teams are busy.

STANDBY - Number of components in operable condition but not in operation due to the maximum allowable number of components already in operation.

STARTSTATS - Starting time for gathering statistics (in corresponding units) for each simulation run.

TCOMPS - Total number of components operating (gathered over several simulation runs).

TOTMCH - Total number of components available in each particular simulation run.

TOTSIM - Total simulation run time in corresponding units for each particular simulation run.

TRPTS - Number of repair teams busy (gathered over several simulation runs).

TRUN - Total number of times a specific set of data is to be tested against different seeds.

TSERVE - Total number of repair teams available for each particular simulation run.

TSTIME - Total statistics gathering time (gathered over several simulation runs).

UNITS - Dimension of parameters (eg, hours, days,...).

ZLAST - Time since last preventive maintenance was performed.

ZX,ZY,ZZ - Contains the random number generation seeds.

C5. Program Maintenance Requirements

The simulation package is divided into three sections. These sections, when compiled, are linked together to form a single file called GPMS.EXE. These sections are listed below. The first section is an object file which should not be edited. The other two, GPMS.FOR and INPUT.FOR, contain the editable Micro-Soft FORTRAN IV code.

CLEAR.OBJ - This contains the object code of an external Assembly Language subroutine. It clears the screen and place the cursor in the upper left-hand corner. The file CLEAR.ASM is the editable code written in Micro-Soft Assembly Language.

GPMS.FOR - This is the main simulation program. Its code performs all simulation functions. This is written in Micro-Soft FORTRAN IV.

INPUT.FOR - This is an external subroutine called by GPMS.FOR. It accepts data from the user through the keyboard and then pass the information to GPMS.FOR. It is written in Micro-Soft FORTRAN IV.

The following files are included in the Micro-Soft FORTRAN IV package. These files are necessary for program compilation.

FOR1.EXE - First pass compiler
PAS2.EXE - Second pass compiler
LINK.EXE - Program linker
LINK.V2
FORTRAN.LIB
FORTRAN.MAP
MATH.LIB (see note *)
MATH.MAP (see note *)

* note: The MATH.LIB and MATH.MAP involve internal numeric representations of data. By renaming a specific Math Library to these names you can tailor your program to a specific computer or a specific internal representation.

The package defaults the Math Library to be compatible with an 8Q86 processor. However, if your computer contains an 8087 processor, you can use it by doing the following:

Rename MATH.LIB and MATH.MAP to 8086.LIB and 8086.MAP respectively (or you may delete them as they are not needed). Then rename 8087.LIB and 8087.MAP included in the package to MATH.LIB and MATH.MAP respectively.

Another Math Library which alters how a program's numeric data is stored is the DECMATH.LIB and DECMATH.MAP. This represents data in decimal form rather than binary. In certain applications this may produce more accurate results. However, the GPMS simulation package is not set up to be compatible with this.

The following is a simple batch file. It contains the necessary instructions to compile and link the various parts of the GPMS package together into the executable GPMS.EXE program.

These files must exist in the directory containing your FORTRAN compiler:-

```
INPUT.FOR
GPMS.FOR
CLEAR.OBJ
```

Commands contained in the batch file:

```
FOR1 INPUT,INPUT,INPUT;
PAS2
FOR1 GPMS,GPMS,GPMS;
PAS2
LINK GPMS+INPUT+CLEAR,GPMS,,,
```

C6. Miscellaneous Notes

C6.1 Data size too large - No error detected but results wrong

The package uses an extensive amount of data space due in part to many large arrays. If more arrays are added or if the size of these arrays are increased, problems may arise. The FORTRAN compiler will not issue an error message. However, strange and absurd results will be produced. One method of solving this problem is to issue the $LARGE meta-command at the very beginning of the two program sections (GPMS.FOR and INPUT.FOR) starting in column 1.

C6.2 Using another version of FORTRAN

Standard FORTRAN only allows variables and routine names up to a maximum of 6 characters in length. Micro-Soft FORTRAN IV allows names to be virtually any size. This program uses several names and data items larger than 6 characters in length. This was done to ease both readability and writeability. These names may have to be changed if using a different version of FORTRAN.

C7. Program Listing

Due to the size of the software package (2200 lines), a listing of the code is not given. If a listing is desired, one may print out the files GPMS.FOR and INPUT.FOR. The first file, GPMS.FOR, contains the code which simulates the system and produces output. The second file, INPUT.FOR, contains the code which accepts all user input.

Below is a listing of a small routine which may be useful to other Micro-Soft FORTRAN IV programs. It is an external Micro-Soft Assembly Language subroutine which, when called, clears the screen and places the cursor in the upper left-hand corner.

```
DATA        SEGMENT PUBLIC 'DATA'
DATA        ENDS
DGROUP      GROUP DATA
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE,DS:DGROUP,SS:DGROUP;
PUBLIC      CLEAR
CLEAR       PROC    FAR
            PUSH    BP
            MOV     BP,SP
            MOV     CX,1
            MOV     AH,0
            MOV     AL,2
            INT     16
            MOV     SP,BP
            POP     BP
            RET     0
            CLEAR   ENDP
            CODE    ENDS
END
```

DISTRIBUTION

Dr. D. Mills	Director of Research
Mr. T.J. Ravishankar	Design & Construction, H14G05
Mr. M. Millicev	Design & Development-Gen, H09A06
Mr. K. Mori	Design & Development-Gen, H11A20
Mr. B.D. MacTavish	Technical & Training Services, A07F06
Mr. W.D. Pantall	Technical & Training Services, A08H5
Mr. H.G. Brandford	Technical & Training Services, A7E1
Mr. S.B. Harvey	Technical & Training Services, A02H01
Mr. F.J. Bradacs	Nanticoke TGS
Mr. J. Higgins	Atikokan TGS
Mr. D.A. McLeod	Lakeview TGS
Mr. K. Oryniak	Lambton TGS
Mr. W.E. Laing	Pickering NGSA
Mr. W.D. McKee	Bruce NGSA
Mr. C. Bromley	Bruce NGSB
Mr. J.G. Cassan	Operations Research
Dr. J. Endrenyi	Operations Research
Dr. L. Wang	Operations Research
Dr. S.H. Sim(4)	Operations Research
Mr. D. Bartlett	Operations Research

Head Office Library
Editorial Section (2)
Office Copies (3)
Extra Copies (2)